STOCK PRICE PREDICTION USING MACHINE LEARNING TECHNIQUES

Machine Learning to Explore Stock Price Prediction

JUNE 1, 2021

Mason Krug

Table of Contents

A. Project Overview

A1. Organizational Need Addressed

My project assisted my firm in the decision-making process needed to analyze a specific stock's value at a specified future date. This was accomplished by automating the process of data gathering and stock analysis. Analysis was conducted utilizing machine learning techniques to predict future stock prices based on historical data.

A2. Project Scope

My firm had implemented a rubric for the projects' deliverables. My project had to automate the process of gathering stock data, perform machine learning analysis of historical data, and output a predicted price for a specific stock with an accompanying graphic. The project would take a total of 4 days to complete. The tasks were broken down into discrete goals with measurable deliverables to validate achievement. The goals were as follows:

- Download data from a stock of choice
- Transform the data into a format easy to interpret and manipulate with Python
- Perform machine learning techniques to output a predicted value
- Display predicted output in a user-friendly manner to aid in decision making.

Each goal and its associated deliverables were successfully achieved.

A3. Solution Overview

When thinking of different approaches to complete this project, I drew on knowledge gained from previous courses. I created a Python application with user-defined functions that download a specific stock's data into a CSV file format. With that CSV file, I performed ETL techniques to prepare the data to be fed into a machine learning algorithm. I chose the Support Vector Machine (SVM) machine learning algorithm as it is suited for time-series-based data. After making the machine learning model, I had it output a line created with the Matplotlib Library in Python to graphically display the prediction. Additionally, predictions were output into the Python terminal window.

B. Summary of Project Execution

B1. Project Plan

The execution of this project had little deviation from the overall scope of the project. One obstacle that I encountered was the application of back up and version control. I re-wrote many different sections of this project and did not want to lose progress that I have made. I decided to use Git and GitHub to host

my project. This was the first time I've used Git and GitHub for a major project. Using GitHub allowed for seamless backups which lessened the fear of data loss. Additionally, its version control features allowed me to peer back into previous versions of my program to avoid having to rewrite specific sections.

B2. Project Planning and Methodology

I largely stuck to the initial methodology that I had established for preparation of this project. The waterfall methodology allowed me to complete discrete components of this project sequentially. I knew that I was able to proceed in development confidently because a previous section had been created.

B3. Project timeline and Milestones

My timeline remained accurate with respect to the number of days to complete. However, the dates that each goal and milestone were accomplished were shifted to the left. I completed this project earlier than expected however it still took the planned amount of time to complete. The milestones and deliverables that I developed were used allowed me to proceed further into development confidentially.

C. Data Selection and Collection Process

C1. How Data Selection and Collection Differed From Original Plans

Stock data is public and free to obtain. However, that does not necessarily mean easy. I needed to gather stock data for a specific stock Ticker from a specified start and end date. I first looked into the Google Finance API to accomplish this. However, interoperability between the Google Finance API and Python is not well documented. Additionally, the Google Finance API seems to be somewhat abandoned by Google. I decided to use the Yahoo! Finance API to gather my stock data. Using the *yfinance* module in Python, I was able to download stock data from any publicly trading stock on the NASDAQ and S&P 500. The *yfinance* module allowed me to specify a start date, end date, and file format to save into. The *yfinance* module ended up fitting my needs and allowed me to complete my project.

C2. Project Planning Methodology

There were no deviations from the methodology that I had planned to use. The waterfall methodology allowed me to focus on discrete objectives. After completing those discrete objectives, I was able to confidently move on to the next deliverable.

C3. Project Timeline and Milestones

As addressed above, there were no deviations in the amount of time it took me to complete my project. However, my project start date was earlier than what I expressed. My actual project start date was on 25 May. This allowed me to complete the project on 29 May. Each milestone was completed on time with respect to the adjusted start date.

D. Extract, Transform, Load Process

D1. Techniques Used and Appropriateness

The cornerstone of this project balanced on the ability to obtain quality, accurate, and complete stock data. Fortunately, *yfinance*, a module in Python built to integrate with the Yahoo! Finance API, was able to provide just that. With an input of a stock ticker, start date, and end date, I was able to download stock data into a CSV file format rather easily. After obtaining the stock data, I used the *Pandas* module in Python to load CSV files into a data frame. I extracted out the features of interest (date and closing price) and parsed them into separate data frames. The Support Vector Machine module would not take date-time format as an input. I used the *toordinal()* function from the *datetime* module to translate a date into an integer The function outputs an integer that increments by 1 from the start date of January, 1st, year 0000. For example, the date "30 May 2021" would be converted into the integer 737940. This translation from date to integer allowed for me to fit my model to stock data ranging from a user-specified "start" date to the date of yesterday.

E. Report on Data Analysis Process

E1. Methods Used to Analyze Data

The primary method I used to analyze my data was the difference, or the variance of the predicted price in comparison to the true price of a stock given on a predicted day. In order to gather this data, I configured my application to predict the current closing price of a stock using historical data as the input for prediction. For example, if the true value of a stock was $100 and the predicted value was $105, then the variance would be positive 5%.

E2. Advantages/Limitations of Data Analysis Techniques

Advantages of this project include an automated way to gather a predicted stock value. It should be noted, however, that historical prices alone are 1 of a myriad of metrics that can influence a stock's price. Trade volume, market beta, current events, natural disasters, supply chain information, and many other variables can influence a stock's price. While it is unreasonable to predict a stock's price 1 year out using historical closing price alone, the margin of error 1 or 2 days into the future is small enough to produce satisfactory results. Future versions of this project will take into account other variables such as trade volume and daily spread to better forecast a price.

E3. Explanation of Analytical Methods

The primary measurement that I used to measure the performance of my application was the variance of a predicted price with respect to its true price. My acceptable limit for this variance is within a 15% boundary greater than and less than the true price. Startups can tend to have high volatility when entering the public market and it is easy for some startups to be significantly overvalued or undervalued. To minimize the exposure to this influence, the stocks I chose to obtain measurements from were from companies within the S&P 500, otherwise known as Blue Chip stocks. Additionally, I gathered data from SPY, the index fund that can be used as a proxy for the S&P 500's overall performance. Running my application produced the following results:

Company	Ticker	Deviation (% from true value)
SPY Index	SPY	3.44%.
Google	GOOG	7.85%
Microsoft	MSFT	2.4%
Tesla	TSLA	-3.82%
Facebook	FB	6.41%
Apple	AAPL	0.22%
Amazon	AMZN	0.82%
Alphabet	GOOGL	6.54%
J. P. Morgan	JPM	3.55%
Johnson and Johnson	JNJ	2.75%
Visa Inc.	V	1.48%
NVidia	NVDA	13.22%
Home Depot	HD	3.07%
Proctor & Gamble	PG	2.02%
Disney	DIS	-4.68%
Bank of America Corp.	BAC	5.38%
Adobe	ADBE	5.7%
Intel Corporation	INTC	-7.35%
Verizon Communications	VZ	-1.66%
Cisco	CSCO	2.92%
Netflix	NFLX	-1.5%
Pfizer Inc.	PFE	2.96%
Coca Cola	KO	3.45%
AT&T	T	-3.33%
Walmart	WMT	3.29%
Average Among All	-	2.21%

Note that the above values were generated from this date rate: 01 Mar 2021 to30 May 2021. The majority of predictions fall well within the 15% margin of error that I had established. The largest outlier, NVidia, has a deviation of 13.22% - still within my established tolerance. In the above table, a negative deviation indicates a predicted price that was less than the true price where a positive deviation indicates a predicted price that was above the true price.

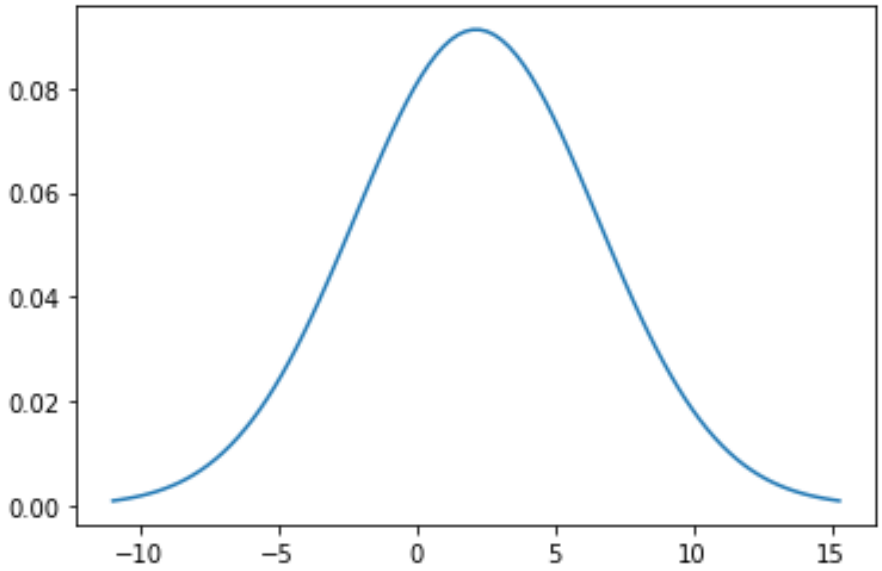
F. Data Evaluation

F1. Evaluation of Statistical Significance of Results

I compiled data from 25 stocks – one of which is an index fund. I remove that as I am interested in only the stock data, not the index funds. With a sample of 24, I was able to calculate the following metrics:

Sample mean of $\bar{x}$ = 2.1636% and sample standard deviation of σ = 4.37504% with a variance of σ^2 = 19.141%.

With those metrics, I can compute the below normal distribution:



The graph is centered on the mean of $\bar{x}$ with a standard deviation of σ. If we can assume that the sample of 24 stocks is indicative of all stocks from within the S&P 500, we can conclude that Considering the 24 stocks as the sample population and if we can assume that the entire population is all the stocks from within the S&P 500, we can calculate that the Z score of a value greater than a 15% deviation is Zupper = 2.93401 and the Z score of a value less than -15% is Zlower = -3.92307. Operating under the assumption that the values from my model are fairly normally distributed, we can conclude that my model producing a value that deviates by greater than 15% is very unlikely. My model produces statistically significant values that are within a 15% margin of error.

F2. Evaluation of Practical Significance of Results

With the accuracy reported above, it is important to note that the historical prices of a stock are 1 of many factors that play into the valuation of a company and its underlying assets. Confidence in prediction greatly reduces when extrapolating well into the future. The practical significance of my model, why statistically significant by one metric, should not be used solely when evaluating a company's performance. However, this tool does meet the requirement of aiding in decision-making when evaluating a company by using historical data as a blueprint for future valuation.

F3. Evaluation of Overall Success and Effectiveness of the Project

In testing my model, I evaluated the accuracy of a handful of different machine learning algorithms that deal with time series. I found SVM to be the most accurate when given a sample of historical prices of less than 1 year. I believe that this tool is an effective way to evaluate 1 of the many metrics that underlie a company's performance. It takes a rote task and simplifies it from the user's point of view.

G. Summary of Analysis

G1. Conclusion Summary

My model is able to predict a stock price within a tolerable deviation of less than 15%. It uses historical closing prices as an input to train a Support Vector Machine model. Once the model is trained, a future date is used as an input to predict a future price. This model alleviates the burden of manually scraping data from web sources and provides a consolidated way to analyze a stock's closing price. With this project, my firm is able to save time and reduce man-hours associated with the aforementioned tasks. This model allows for one aspect of stock analysis to be automated.

G2. Explanation of Chosen Tools and Graphical Representation

The use of Python for this project was an easy decision. Using Python allowed me to build on previous knowledge that I had gained in pursuit of my degree. Additionally, the extensibility and user base surrounding Python provided the ability to create an application that met my needs. Since my application forecasts a predicted stock value, the use of charts to display that value was a suitable choice. The Matplotlib library allows for the creation of charts using the data my program provided. This allowed for trends to be easily visualized. Moreover, my program outputs a text summary detailing the predicted stock price. With both a graphical representation of a price and a text output, my program is able to effectively illustrate and convey to the user what it predicts a future value to be.

G3. Courses of Action Recommendation

Two further improve my model I can recommend the addition of two additional features to add. The first is the inclusion of trade volume as a metric to analyze. Trade volume is the number of completed trades over a given period. The period of time that makes the most sense is trading days. In other words, how much of a given stock is traded per trading day? This can be used as a proxy for a stock's popularity and its volatility. A stock that is more frequently traded is more likely to have higher volatility. The second feature I would add to improve my model would be to include intraday high and low values as an input into my machine learning model. A stock with a large spread between its intraday high and low values would have a larger change to deviate from a current value during the next trading day.

Of course, these are 2 of many features that could be added to my model. Future additions could also include the addition of natural disasters and world events from news sources. For example, a program that could scrape information from a news article about an oil spill in the Gulf Coast could be used to recommend a short position on oil-related companies. Moreover, events such as a merger or takeover in a company tend to significantly affect a stock price in the short term. A web scraper that would collect this information, analyze it, and interpret it as either positive or negative could be fed into my model to further increase accuracy and precision.

H. Panopto Recording

H1. Link to Recording

Panopto Recording can be found here:
https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=f415b831-677b-4fde-b13f-ad3d0002ba8b

I. Appendices

I1. Project Source - GitHub

Source code for this project is available here: https://github.com/mkrug6/Capstone

I2. File and Directory Description

File or Directory	Description
main.py	Main function to download, analyze, predict, and save stock data
plot.py	Contains functions to generate and save graphs
metrics.py	Used to generate metrics about the predicted price, such as deviation
svm.py	Contains the functions to create and use the SVM model
config.py	Contains global configurations, such as start and end date, and tickers
download.py	Function to download stock data as CSV
Data Directory	Contains the CSV files downloaded from download.py
Figures Directory	Figures generated from plot.py that contain predicted stock information
Writing Directory	Contains the Task 1, 2, and 3 submissions